REMARKS

Reconsideration of the present application is respectfully requested in view of the following remarks. Prior to entry of this response, Claims 1-24 were pending in the application, of which Claims 1, 23, and 24 are independent. In the Final Office Action dated May 17, 2004, Claims 1-24 were rejected under 35 U.S.C. § 102(b) and Claims 1, 23, and 24 were rejected under 35 U.S.C. § 112. Applicant hereby addresses the Examiner's rejections in turn.

I. Rejection of the Claims Under 35 U.S.C. §112, First Paragraph

In the Final Office Action dated May 17, 2004, the Examiner rejected Claims 1, 23, and 24 under 35 U.S.C. § 112, first paragraph, as containing subject matter which was not described in the specification in such a way to reasonably convey to one skilled in the art, at the time the application was filed, that the inventor had possession of the claimed invention. Applicant respectfully traverses this rejection.

The Examiner states that the specification does not contain sufficient support for Claims 1, 23, and 24. (*See* Final Office Action, page 3, lines 4-5.) In contrast, Applicant submits that support for "moving an instruction sequence speculatively executable forward an instruction sequence not speculatively executable in the program" can be found in the specification as filed at least at: i) line 7 on page 50 to line 23 on page 52; ii) line 7 on page 57 to line 19 on page 57; and iii) FIG. 23 – FIG. 25.

Next, the Examiner states that "a non-speculative instruction is executed before a speculative instruction" on pages 24 and 60 is not inconsistent with the recitation "moving an instruction sequence speculatively executable forward an instruction sequence not speculatively executable." In contrast, Applicant states these are not inconsistent. The reason for this is explained as follows.

With embodiments of the present invention, for example, when a program is complied with a compiler, the compiler cannot predict a decision result of a commit instruction (e.g. a branch instruction) in the program. For example, the compiler cannot discriminate between: i) an instruction sequence having execution result that may be used; and ii) an instruction sequence having an execution result that may be abandoned. Accordingly, the compiler regards these instructions, for example, as a speculative instruction sequence and moves the speculative instruction sequence forward a non-speculative instruction sequence in the program.

Furthermore, consistent with an embodiment of the invention, each instruction sequence in a program is inserted into one queue (e.g. Nos. 1-4 of FIG. 5) of one execution buffer (e.g. Nos. 1-3 of FIG. 5) that corresponds to an address (location) and a task number of the instruction sequence. The queue "Nos. 1-4" represents the task number of the instruction, and the execution buffer "Nos. 1-3" represents the address (location) of the instruction. The speculative instruction sequence is inserted into the queue in comparison with the non-speculative instruction sequence because, for example, the speculative instruction sequence was moved forward the non-speculative instruction sequence in the program. In general, however, an execution unit's processing speed (*See* elements 12, 13, and 14 of FIG. 3) is slower than each instruction's transfer speed from the queue to the execution unit. Accordingly, as time goes by, a plurality of instruction sequences of the same address are respectively stored in different queues Nos. 1-4 of the same execution buffer corresponding to the same address.

On the other hand, as shown in FIGs. 3 and 5, an operand condition decision unit 11 decides which instruction stored at a head position of each queue Nos. 1-4 in the same execution buffer can be preferentially executed. For example, as for the execution buffer No. 1, one instruction is selected from four instructions stored at each head position of four queues Nos. 1-4, and then transferred to the execution unit No. 1. However, each instruction stored at a head position of each queue Nos. 1-4 in the same execution buffer is not limited to the speculative instruction sequence (moved forward). As mentioned above, as time goes by, the non-speculative instruction sequence (not moved forward) having the same address as the speculative instruction sequence is often stored at a head position of a different queue in the same execution buffer. In this case, the non-speculative instruction is preferentially transferred to the execution unit rather than the speculative instruction. This is because an execution result of the non-speculative instruction is certainly used whereas an execution result of the speculative instruction may be abandoned. Accordingly, "a non-speculative instruction is executed before a speculative instruction" on pages 24 and 60 means that "the non-speculative instruction is preferentially transferred to the execution unit rather than the speculative instruction."

Furthermore, "a non-speculative instruction is executed before a speculative instruction" represents an execution stage of a program by an execution unit. On the other hand, "moving an instruction sequence speculatively executable forward an instruction sequence not speculatively executable" represents a compiling stage of a compiler and not an execution stage of a program. Briefly, the two stages are different. Accordingly, these two sentences are not inconsistent.

II. Rejection of the Claims Under 35 U.S.C. § 112, Second Paragraph

In the Final Office Action, the Examiner rejected Claim 2 under 35 U.S.C. § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the Applicants regard as their invention. Claim 2 has been amended, and Applicant respectfully submits that the amendment overcomes this rejection and adds no new matter.

III. Rejection of the Claims Under 35 U.S.C. § 102(b)

In the Final Office Action, the Examiner rejected Claims 1-24 under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent. No. 5,511,172 ("*Kimura*"). Applicant respectfully traverses this rejection.

Generally, execution of a program by a processor includes the following five stages:

(a) The program is stored in a memory of the processor;

(b) Each instruction of the program is fetched from the memory in order;

(c) Each instruction of the program is decoded in order;

(d) Each instruction is differently transferred to one of a plurality of execution units in the processor; and

(e) Each instruction is executed by one execution unit in the processor.

In the first place, a feature of the claimed invention is directed to program scheduling by a compiler. The program scheduling is executed before the above stage (a). Consequently, the present invention as claimed is not directed to executing a program by a processor.

Next, program scheduling by a compiler, consistent with an embodiment of the present invention, is explained as follows. A program (e.g. shown in FIG. 23) is divided into a plurality of instruction sequences (e.g. data dependence sequences 1-11 in FIG. 24). Each instruction sequence comprises a plurality of instructions not executable in parallel.

Regarding an embodiment of the present invention, data dependency between two instruction sequences is decided in instruction sequences. For example, an instruction sequence having data dependency with another instruction sequence, and an instruction sequence not having data dependency with another instruction sequence are discriminately decided.

An instruction sequence speculatively executable, consistent with an embodiment of the present invention, is moved forward an instruction sequence not speculatively executable on the program, for example, as shown in FIG. 25. An instruction sequence speculatively executable is, for example, an instruction sequence not having data dependency with another instruction sequence, and the instruction sequence not speculatively executable is an instruction sequence having data dependency with another instruction sequence.

Consistent with an embodiment of the present invention, at a program's compile time, an instruction sequence not having data dependency with another instruction sequence is moved forward an instruction sequence having data dependency with another instruction sequence. As a result, the instruction sequence not having data dependency with another instruction sequence is speculatively fetched and decoded rather than an instruction sequence having data dependency with another instruction sequence. (*See*, for example, stages (b) (c) above.) Accordingly, at the program's execution time (e.g. above steps (d) (e)), the program as complied above can be more quickly executed because the instruction sequence speculatively fetched and decoded does not have data dependency with another instruction sequence.

Kimura discloses that when a conditional branch instruction in a program is decoded, a type of the conditional branch instruction is decided. If branching of the conditional branch instruction has a high possibility, an instruction sequence to be executed after branching is preferentially fetched from the program, decoded, and transferred to one execution unit. If non-branching of the conditional branch instruction has a high possibility, a succeeding instruction sequence to be executed following the conditional branch instruction is preferentially fetched from the program, decoded, and transferred to one execution unit. If branching and non-branching of the conditional branch instruction have respectively an equal possibility, the instruction sequence to be executed after branching and the succeeding instruction sequence to be executed following the conditional branch instruction are respectively fetched from the program, decoded, and transferred to two execution units.

In contrast with the claimed invention, *Kimura* is directed to execution of a program by a processor (e.g. the above stages (a) – (e)). *Kimura* does not teach, suggest, or disclosed program scheduling by a compiler. Moreover, *Kimura* does not teach dividing the program into a plurality of instruction sequences on the condition that each instruction sequence comprises a plurality of instructions not executable in parallel. Furthermore, data dependency between every two instruction sequences is not decided in all instruction sequences in *Kimura*.

As mentioned above, an instruction sequence to be executed after branching or a succeeding instruction sequence to be executed following the conditional branch instruction is preferentially fetched and decoded (e.g. stages (b) (c) above.) With *Kimura*, however, it is possible that the instruction sequence after branching or the succeeding instruction sequence has data dependency with another instruction sequence. In this case, when the instruction sequence after branching or the succeeding instruction sequence is preferentially fetched and decoded, in *Kimura*, it is possible that another instruction sequence is executed by one of a plurality of execution units. In spite of other execution units being idle, the instruction sequence after branching or the succeeding instruction sequence cannot be transferred to the other execution unit until execution of another instruction sequence is completed. Accordingly, the program taught by *Kimura* cannot be quickly executed because it is possible that the instruction sequence preferentially fetched and decoded has data dependency with another instruction sequence.

In sum, *Kimura* does not anticipate the claimed invention because Kimura at least does not disclose moving an instruction sequence speculatively executable forward an instruction sequence not speculatively executable in the program, as recited by amended Claim 23. Amended Claims 1 and 24 include similar recitations. Accordingly, independent Claims 1, 23, and 24 patentably distinguish the present invention over the cited art, and Applicant respectfully requests withdrawal of this rejection of Claims 1, 23, and 24.

Dependent Claims 2-22 are also allowable at least for the reasons above regarding independent Claim 1, and by virtue of their dependency upon independent Claim 1. Accordingly, Applicant respectfully requests withdrawal of the rejection of dependent Claims 2-22.

IV. Conclusion

Applicant respectfully requests that this Amendment After Final be entered by the Examiner, placing the claims in condition for allowance. Applicant respectfully submits that the proposed amendments of the claims do not raise new issues or necessitate the undertaking of any additional search of the art by the Examiner, since all of the elements and their relationships claimed were either earlier claimed or inherent in the claims as examined. Therefore, this Amendment should allow for immediate action by the Examiner.

Finally, Applicant respectfully submits that the entry of the Amendment would place the application in better form for appeal, should the Examiner dispute the patentability of the pending claims.

In view of the foregoing remarks, Applicant respectfully submits that the claimed invention, as amended, is neither anticipated nor rendered obvious in view of the prior art references cited against this application. Applicant therefore requests the entry of this Amendment, the Examiner's reconsideration and reexamination of the application, and the timely allowance of the pending claims.

In view of the foregoing, Applicant respectfully submits that the pending claims, as amended, are patentable over the cited references. The preceding arguments are based only on the arguments in the Official Action, and therefore do not address patentable aspects of the invention that were not addressed by the Examiner in the Official Action. The claims may include other elements that are not shown, taught, or suggested by the cited art. Accordingly, the preceding argument in favor of patentability is advanced without prejudice to other bases of patentability.

Please grant any extensions of time required to enter this amendment and charge any additional required fees to our Deposit Account No. 06-0916.

Respectfully submitted,

FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, L.L.P.

Dated: July 27, 2004

By: [illegible]
D. Kent Stier
Reg. No. 50,640
(404) 653-6559